UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/817,880	03/26/2001	Sean E. Trowbridge	167381.01	7613

47973 7590 10/22/2009

WORKMAN NYDEGGER/MICROSOFT
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UT 84111

EXAMINER	
RUTTEN, JAMES D	
ART UNIT	PAPER NUMBER
2192	

MAIL DATE	DELIVERY MODE
10/22/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte SEAN E. TROWBRIDGE

Appeal 2009-004409
Application 09/817,880[1]
Technology Center 2100

Decided: October 22, 2009

Before JAMES D. THOMAS, JAY P. LUCAS, and JOHN A. JEFFERY, *Administrative Patent Judges*.

LUCAS, *Administrative Patent Judge*.

DECISION ON APPEAL

---

[1] Application filed March 26, 2001. The real party in interest is Microsoft Corporation.

STATEMENT OF THE CASE

Appellant appeals from a final rejection of claims 1 to 33 under authority of 35 U.S.C. § 134(a). The Board of Patent Appeals and Interferences (BPAI) has jurisdiction under 35 U.S.C. § 6(b).

Appellant's invention relates to a method and system for generating executable code for operating a computer. In the words of Appellant:

> In order to address … intermediate language compilers have been developed that convert source code written in many different languages *(e.g.,* C++, BASIC, Pascal) to an intermediate language that operates in a virtual environment sometimes referred to as a virtual machine. This provides a layer of abstraction to the source code that enables a single code source to execute on multiple hardware platforms and operating environments. Thus, the virtual machine converts the intermediate language into the particular instructions required of the platform. Although platform independence and reduced development time is generally achieved by this model, some program performance is sacrificed when converting the intermediate language to machine-dependent instructions. Thus, a tradeoff exists between higher performance program operations provided by standard native compilers and the compilation and development time benefits provided by intermediate language systems. Consequently, there is a need for a system and methodology to provide program performance approaching that of native compiled systems while also providing the development benefits of intermediate language systems.
>
> . . . .
>
> More particularly, the present invention stores and/or caches possible executable images associated with at least one operating environment in a repository, such as an image database. A loader searches the repository for a possible executable image at run time in a virtual execution environment. If the loader finds a suitable image, the image is loaded and executed by a virtual code execution engine. If a suitable image is not found by the loader, a logging component

> logs information about the current execution environment according to the operating conditions of the execution engine. The logged information is provided as feedback and analyzed either manually and/or automatically to generate a new and/or specialized executable image that may then be stored in the repository and utilized in future operations of the execution engine. The specialized image generally provides higher performance operations within a virtual system in contrast to generating native code anew from a generic intermediate language according to operating environment differences. Analysis for generating the specialized executable may include operating environment considerations such as hardware configurations, operating system versions, processor architectures and configurations, component versions, developer parameters, domain determinations, security considerations, along with a plurality of other factors.

(Spec. 2, ll. 5-20; Spec. 3, ll. 13-28.)

Claim 1 and claim 30 are exemplary:

> 1. A computer implemented system for generating specialized executables, comprising the following computer executable components:
>
> a virtual subsystem that processes a generic code image;
>
> a loader to determine availability of a specialized image that is associated with an operating environment of the virtual subsystem; and
>
> a log to store generic code image runtime information relating to the operating environment of the virtual subsystem, the runtime logged information includes at least a set of information related to a particular user to create a native executable according to the particular user, the logged information is employed as feedback to generate the native executable based upon the availability of the specialized image,

the native executable is utilized to provide improved performance of the virtual subsystem

30. A computer implemented signal transmitted between at least two processes executing on one or more computers facilitating generation of specialized executables, comprising:

a signal for communicating between one or more components of a virtual system, the virtual system processing a generic code image and logging information relating to an operating environment and a particular user of the virtual system *via* the signal, the logged information includes at least a set of information to create a specialized native executable according to the particular user;

wherein the logged information is employed as feedback across the signal to generate a specialized native executable if the generic code image is determined incompatible with the operating environment of the virtual system, the specialized native executable is utilized to provide improved performance of the virtual system.

The prior art relied upon by the Examiner in rejecting the claims on appeal is:

Breslau	US 5,761,512	Jun. 02, 1998
Knight	US 6,126,330	Oct. 03, 2000
Goodwin	US 6,158,049	Dec. 05, 2000
Nelin	US 6,253,368	Jun. 26, 2001
Ramezani	US 6,457,122 B1	Sep. 24, 2002
Spyker	US 6,571,389	May 27, 2003
Fogarty	US 6,721,946	Apr. 13, 2004

Armstrong, *HotSpot: A new breed of virtual machine*, JavaWorld, March 1998, http://www.javaworld.com.

Aho, Compilers: Principles, Techniques, and Tools 1-24 (Michael A. Harrison ed., Addison Wesley 1988) (1986).

REJECTIONS

The Examiner rejects the claims as follows:

R1: Claims 1, 2, 5-17, 19, 30 and 31 stand rejected under 35 U.S.C. § 103(a) for being obvious over Breslau, Spyker, Armstrong and Knight.

R2: Claims 3 and 4 stand rejected under 35 U.S.C. § 103(a) for being obvious over Breslau, Spyker, Armstrong and Knight and further in view of Fogarty.

R3: Claim 18 stands rejected under 35 U.S.C. § 103(a) for being obvious over Breslau, Spyker, Armstrong and Knight and further in view of Nelin.

R4: Claims 20-22 and 27-29 stand rejected under 35 U.S.C. § 103(a) for being obvious over Goodwin in view of Knight.

R5: Claims 23 and 24 stand rejected under 35 U.S.C. § 103(a) for being obvious over Goodwin and Knight and further in view of Aho.

R6: Claims 25 and 26 stand rejected under 35 U.S.C. § 103(a) for being obvious over Goodwin, Knight, Aho, and further in view of Breslau.

R7: Claim 32 stands rejected under 35 U.S.C. § 103(a) for being obvious over Breslau in view of Spyker, Knight and Nelin.

R8: Claim 33 stands rejected under 35 U.S.C. § 103(a) for being obvious over Ramezani, Knight and Spyker.

R9: Claims 19, 30 and 31 stand rejected under 35 U.S.C. § 101 because the claimed invention is directed to non-statutory subject matter (signals).

Concerning rejection [R9] under 35 U.S.C. § 101, the Examiner has mistakenly stated that the rejection is not under review as the rejection has not been argued in the Appellant's Brief. (Ans. 4). Mere failure to argue a rejection does not negate the appeal of that rejection (MPEP 1205.02, para.

2). Thus, the rejection [R9] of the affected claims 19, 30 and 31 will be considered, and the rejection will be affirmed *pro forma* (summarily sustained) because the rejection has not been traversed by Appellant. It should be pointed out that the rejection of these claims, addressed to a signal, is consistent with the holding of *In re Nuijten*, 500 F.3d 1346, 1359 (Fed. Cir. 2007), *en banc denied,* 515 F.3d 1361 (Fed. Cir. 2008), *cert. denied*, 129 S.Ct. 70 (2008).

Groups of Claims:

The rejections will be discussed in the order of the arguments.

See 37 C.F.R. § 41.37 (c) (vii). *See also In re McDaniel*, 293 F.3d 1379, 1383 (Fed. Cir. 2002) ("If the brief fails to meet either requirement [of 37 C.F.R. § 1.192(c)(7)], the Board is free to select a single claim from each group of claims subject to a common ground of rejection as representative of all claims in that group and to decide the appeal of that rejection based solely on the selected representative claim.").

Appellant contends that the claimed subject matter is not rendered obvious by the combinations of references for failure of the references to teach the specific limitations of the claims. The Examiner contends that each of the claims is properly rejected.

Rather than repeat the arguments of Appellant or the Examiner, we make reference to the Briefs and the Answer for their respective details. Only those arguments actually made by Appellant have been considered in this opinion. Arguments that Appellant could have made but chose not to make in the Briefs have not been considered and are deemed to be waived.

We affirm the rejections.

ISSUE

The issue is whether Appellant has shown that the Examiner erred in rejecting the claims under 35 U.S.C. § 103(a). The issue turns on whether the combination of references teaches the logging of generic code image runtime information including a set of information related to a particular user, in the manner claimed.

FINDINGS OF FACT

The record supports the following findings of fact (FF) by a preponderance of the evidence:

1. Appellant has invented a system and method for generating computer code executable by specific target computer configurations (Spec. 5, l. 15). Rather than develop the code from scratch, the system starts with a generic set of codes written in an intermediate language (Fig. 2, #60). These are executed in a virtual machine environment by execution engine #58, but only after first searching a library of native images from repository #64 for a native image that specifically matches the runtime characteristics of the required virtual environment (Spec. 7, ll. 1-11). If the execution engine finds an already created native image that matches the required characteristics of operating system, processor type, virtual system version and so forth (*See* Fig. 6), then that image is used as it will execute faster than the more generically compiled intermediate language image. But if the system must use the generic image, then it will be endured and customized by a feedback loop involving the logging of operating characteristics of the compiling of the generic code with an associated set of environmental information relating to the current runtime configuration (Spec. 5, l. 9). The images may also be

specialized according to a particular user, for example the number of times in a given period that a particular image is loaded, perhaps by a particular user (Spec. 6, l. 20).

2. The references Breslau, Armstrong, Spyker and Knight all address aspects of the generation of computer code for execution on a general purpose computer.

3. [Specific further findings of fact will be cited in the Analysis section below as part of the analysis therein.]

PRINCIPLES OF LAW

Appellants have the burden on appeal to the Board to demonstrate error in the Examiner's position. *See In re Kahn*, 441 F.3d 977, 985-86 (Fed. Cir. 2006) ("On appeal to the Board, an applicant can overcome a rejection [under § 103] by showing insufficient evidence of prima facie obviousness or by rebutting the prima facie case with evidence of secondary indicia of nonobviousness.") (quoting *In re Rouffet*, 149 F.3d 1350, 1355 (Fed. Cir. 1998)).

"It is well established that '[t]he first door which must be opened on the difficult path to patentability is § 101." *State St. Bank & Trust Co. v. Signature Fin. Group, Inc.,* 149 F.3d 1368, 1372 n.2 (Fed. Cir. 1998) (quoting *In re Bergy,* 596 F.2d 952, 960 (CCPA 1979)) (quoted in *In re Comiskey,* 499 F.3d 1365, 1371 (Fed. Cir. 2007) ("Only if the requirements

of § 101 are satisfied is the inventor 'allowed to pass through to' the other requirements for patentability."))

ANALYSIS

From our review of the administrative record, we find that the Examiner has presented a prima facie case for the rejections of Appellant's claims under 35 U.S.C. § 103(a). The prima facie case is presented on pages 6 to 19 of the Examiner's Answer. In opposition, Appellant presents a number of arguments.

Arguments with respect to the rejection of claims 1, 2, 5 to 17 and 19, 30 and 31[2] under 35 U.S.C. § 103(a) [R1]

Claims 19, 30 and 31 are subject to a non-argued rejection under 35 U.S.C. § 101 [R9] which we have sustained *pro forma* above. As statutory subject matter is a "first door" consideration for patentability, barred at the threshold, the analysis of these claims under 35 U.S.C. § 103(a) is not necessary. (*See In re Comiskey*, 499 F.3d at 1371.)

Concerning the remaining claims, Appellant first argues that "The cited art does not disclose or suggest how this table [in Breslau, a table that equates affinity values to runtime environments,] is populated and

[2] Claim 31 states that it depends from itself. This is obviously a typographical error, and we presume that Appellants intended for claim 31 to depend from claim 30.

specifically is silent regarding logging information during runtime execution of the compiled image as in applicant's claimed invention." (App. Br. 7, middle).

Breslau teaches a computer compiler system that has multiple execution environments (Abstract). The unique execution environment characteristics that define a class of objects to be compiled are called affinities in the Breslau patent (Col. 3, l. 13). Affinities include such items as the operating system of the environment, the processor, the user interface, etc. (Fig. 2; Col. 4, l. 28-30). The characteristics of the execution environment associated with the compiling of a class of objects are listed in the Affinity Resource Table (Col. 4, l. 63). A subset of the table for the specific environment being compiled is listed in the System Environment Table (SET) that includes such items as the processor type and operating system used for the specific instantiated class of objects (Fig. 2; col. 5, l. 17). If there is a specific set of affinities listed for the compiling of a certain set of objects, then a customized compilation takes place; if not, then the compiling is performed for any available execution environment (Col. 5, l. 55). This teaching is read by the Examiner on the use of a specific native executable if available and if not on the claimed generic code image (Ans. 8, middle). Armstrong reinforces that teaching by disclosing "When a method is invoked, the native machine-code version is used, if it exists. Otherwise the bytecodes are reinterpreted." (Armstrong 4, top). Spyker likewise teaches the execution of a runtime image, under user control, on a virtual subsystem (Col. 14, ll. 44-53).

In answer to Appellant's argument above, Examiner states that Knight was cited to teach the information logged during the execution of the application, not Breslau, Spyker or Armstrong, although they are all in the

analogous art of compiling computer objects for multiple environments (Ans. 19-20). Rather, indicates the Examiner, Knight was cited for teaching the feedback, during the runtime execution of a program, of information related to the use by a particular user (Ans. 20, bottom). Examiner points to Knight (Col. 6, ll. 22-27) for the teaching. We find Knight does indeed teach a user interface that indicates object identifiers "being changed, affected by or interacted with in some way by the developer/user, are logged in object display area 15 of the screen of setup tool 10." (Id.).

We thus agree with the Examiner that the cited references do teach the claimed log to store runtime information, which information relates to a particular user which is employed as feedback. That feedback is used to generate customized, native executables (Knight col. 6, ll. 39-43; col. 8, ll. 1-6).

Appellant argues that Knight does not teach runtime logged feedback from the user used to generate a native executable (App. Br. 10, top and bottom; 11, middle). We agree with the Examiner (Answer 21, bottom) that Knight does indeed teach the claimed limitation for the reasons cited by the Examiner on that point.

We, thus, are not persuaded of error in the Examiner's rejection [R1].

Arguments with respect to the rejections
of all other claims
under 35 U.S.C. § 103 [R2 to R8]

All of Appellant's other arguments for rejections [R2] to [R8] are based on the alleged deficiency of the cited art to indicate the feedback based on information related to a particular user to create a native executable

(App. Br. 10-13). As we do not agree that such a deficiency has been established, we are not persuaded of error in these rejections.

CONCLUSIONS OF LAW

Based on the findings of facts and analysis above, we conclude that the Examiner did not err in rejecting claims 1-33 under rejections [R1] to [R9].

DECISION

The Examiner's rejections of claims 1-33 under rejections [R1] to [R9] are Affirmed.

No time period for taking any subsequent action in connection with this appeal may be extended under 37 C.F.R. § 1.136(a)(1)(iv).

AFFIRMED

peb

WORKMAN NYDEGGER/MICROSOFT
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UT 84111